Methods:

Evaluating Code in Different Contexts

This video presents methods from Standard Ruby Library that allow you to evaluate code in different contexts. This is an essential part of metaprogramming in ruby.

Scopes where you can evaluate:

- Instance scope
  (instance_eval, instance_exec)
- Class/module scope
  (class_eval, class_exec, module_eval, module_exec)

instance_eval / instance_exec

```ruby
class KlassWithSecret
    def initialize
        @secret = 99
    end

    private
    def the_secret
        "Ssssh! The secret is #{@secret}."
    end
end

k = KlassWithSecret.new

k.instance_eval { @secret }                    #=> 99
k.instance_eval { the_secret }                 #=> "Ssssh! The secret is 99."
k.instance_eval { |obj| obj == self }          #=> true
```

instance_eval / instance_exec

```ruby
class KlassWithSecret
    def initialize
        @secret = 99
    end
end

k = KlassWithSecret.new

k.instance_exec(5) { |x| @secret + x }          #=> 104


foo = 42

k.instance_eval { @secret + foo }               #=> 141
```

class_eval / class_exec

```ruby
# Ruby documentation:
#   http://ruby-doc.org/core/Module.html#method-i-class_exec

class Thing
end

Thing.class_exec do
    def hello()
        "Hello there!"
    end
end

puts Thing.new.hello()
```

```ruby
#   Thing.module_exec do     <-- module_exec()
#       def hello()
#           "Hello there!"
#     end
#   end

# "Hello there!"
```

Evaluating string code with eval

```ruby
# Ruby documentation:
#   http://ruby-doc.org/core/Kernel.html#method-i-eval

def get_binding(str)
    return binding
end

str = "hello"

eval "str + ' Fred'"                          #=> "hello Fred"
eval "str + ' Fred'", get_binding("bye")      #=> "bye Fred"
```

Calling methods: send, public_send

```ruby
class MyClass
    def foo
        "public foo"
    end

    private
    def bar
        "private bar"
    end
end

obj = MyClass.new

obj.send(:foo)                          # "public foo"
obj.send(:bar)                          # "private bar"

obj.bar                                 # NoMethodError: private method `bar' called for #<MyClass:0x007f84e

obj.public_send(:foo)                   # "public foo"
obj.public_send(:bar)                   # NoMethodError: private method `bar' called for #<MyClass:0x007f84e

obj.__send__(:bar)                      # "private bar"
```